

データマイニングと情報可視化

Week 5

稲垣 紫緒

いながき しお

理学研究院 物理学部門 / 共創学部

inagaki@phys.kyushu-u.ac.jp

ウェスト1号館 W1-A823号室

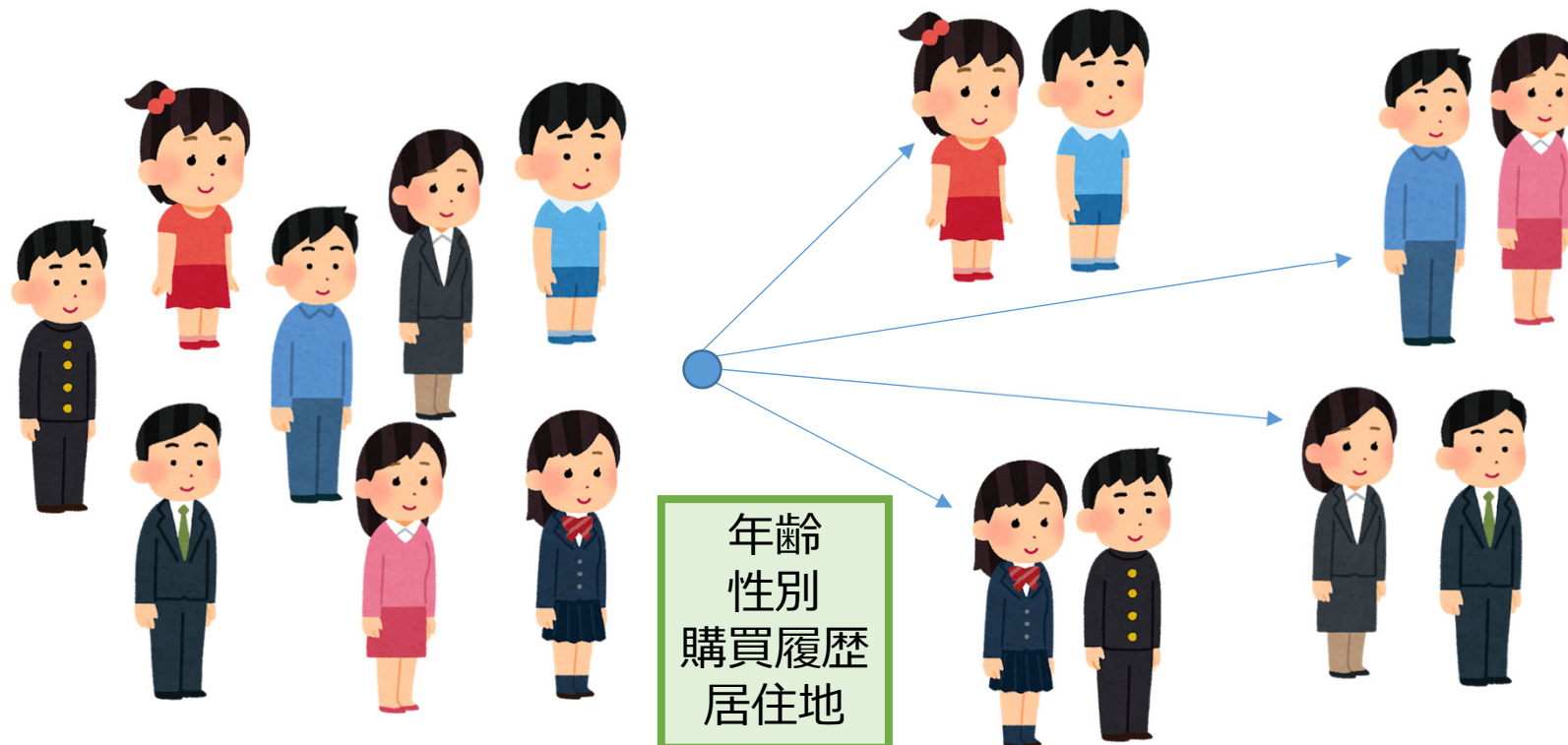
授業計画



データマイニングの代表的な手法

(2) クラスタ分析

似ているデータごとにデータをまとめて分類
→適切な商品を推奨できる

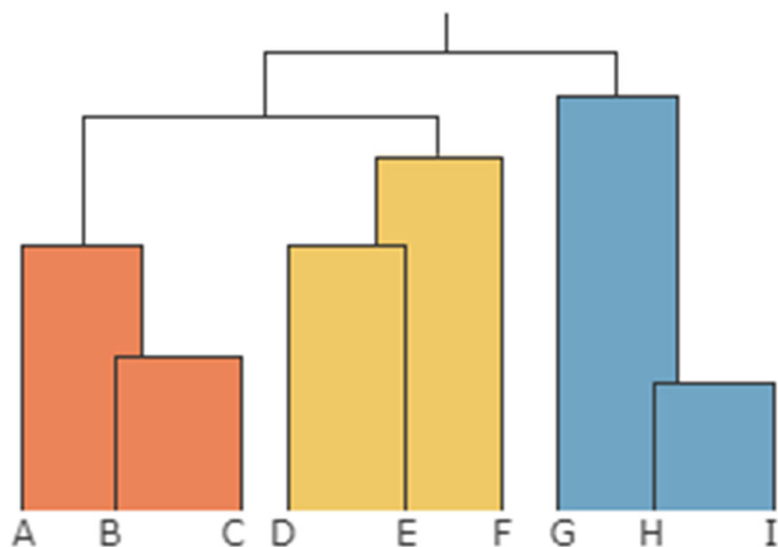


クラスター分析

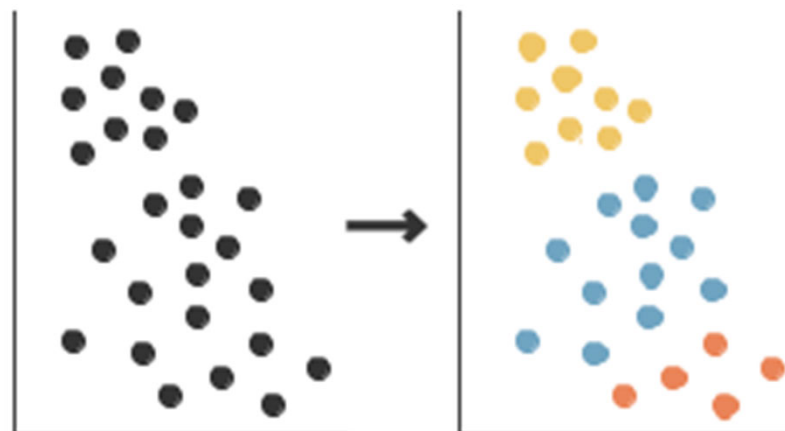
クラスター=Cluster
→房、集団、群れ

教師なし機械学習の一種
いくつかのクラスターになるべきか、
といった答えはない。

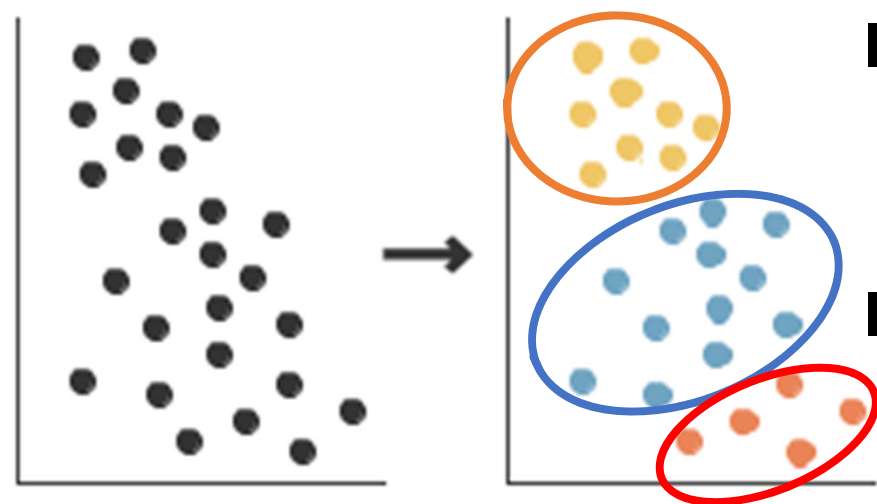
階層型クラスター分析



非階層型クラスター分析



非階層型クラスター解析



- k-means法

- 混合ガウスモデル
(Gaussian Mixture Model, GMM)

- EM アルゴリズム
(Expectation-Maximization Algorithm)

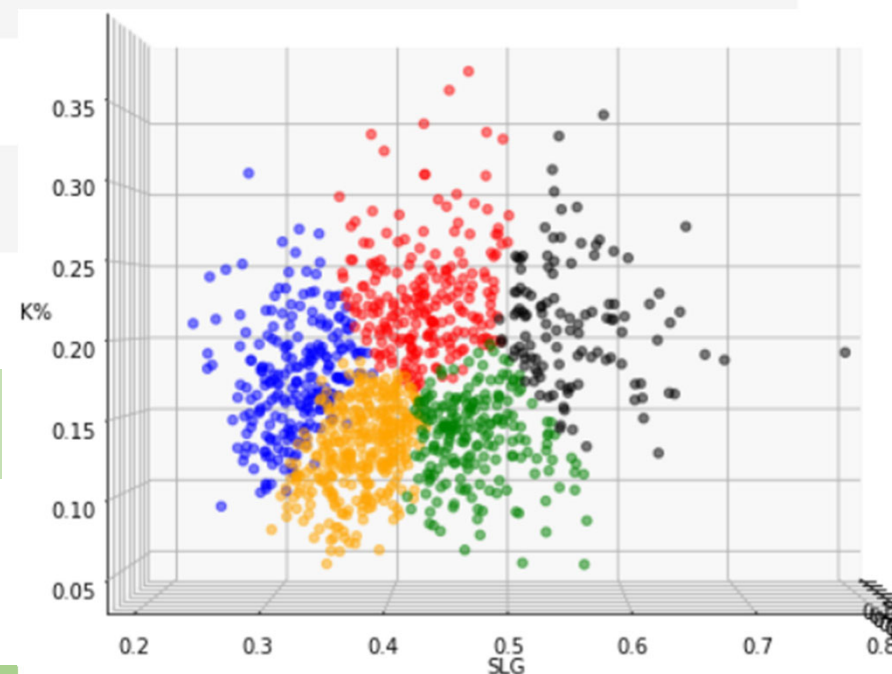
クラスタ分析の適用例

<https://bb-multi-tech.com/programing/scikit-learnk-means-clustering-npb/>


順位	選手名	チーム	打率	試合	打席数	打数	得点	安打	二塁打	...	犠打	犠飛	四球	敬遠	死球	三振	併殺打	出塁率	長打率	year
0	1	今成 亮太	日本ハム	1.000	1	1	1	1	1	...	0	0	0	0	0	0	0	1.000	2.000	2010
1	1	渡部 龍一	日本ハム	1.000	1	1	1	0	1	...	0	0	0	0	0	0	0	1.000	1.000	2010
2	3	金澤 岳	ロッテ	0.571	6	7	7	0	4	...	0	0	0	0	0	3	0	0.571	0.571	2010
3	4	青木 宣親	ヤクルト	0.358	144	667	583	92	209	...	44
4	5	平野 恵一	阪神	0.350	139	593	492	77	172	...	22

5 rows × 26 columns

野球選手の能力を分類




30変数ある



恶性

クラスター解析の手順



(1) グループ分けの対象

サンプルを分類するのか、変数を分類するのか

(2) 分類の形式（種類、生成）

階層的方法か非階層的方法か

(3) 分類に用いる対象間の距離（類似度）

ユークリッド距離、マハラノビス距離、
コサイン距離 など

(4) クラスターの合併方法 (クラスター間の距離の測定方法)

ウォード法、群平均法、最短距離法、最長距離法など

k-means法



nstep=9

クラスタ数=5



step 0 クラスタの数を決める

step 1
各点にランダムにクラスタを割り当てる

step 2
クラスタの重心を計算

変化あり
2に戻る

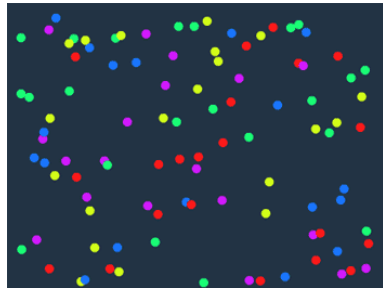
step 3
点のクラスタを、
一番近い重心のクラスタに変更する

クラスタの
組み換えなし

終了

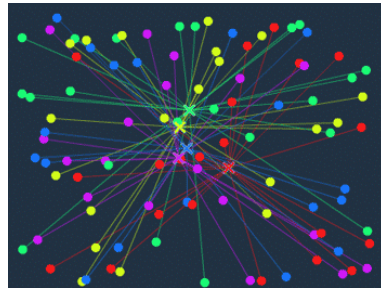
k-means法

クラスタが球形であり、
データのばらつきが等しい



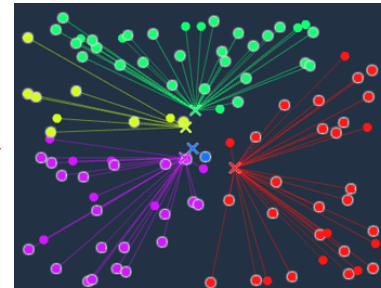
nstep=1

ランダムに割り当て



nstep=2

クラスタの
重心を計算



nstep=3

一番近い重心
のクラスタに
割り当て直し



nstep=4

クラスタの
重心を計算



nstep=5

一番近い重心
のクラスタに
割り当て直し



nstep=6

クラスタの
重心を計算



nstep=7


一番近い重心
のクラスタに
割り当て直し



nstep=8

クラスタの
重心を計算

サンプル間の距離



距離(distance) / 非類似度(dissimilarity)
データやクラスタの似ていなさ

距離の公理

- (1) 距離はマイナスにはならない
 - (2) 同一であれば距離はゼロ
 - (3) 2つの距離はどちらから測っても同じ
 - (4) 三角形の2辺の距離の合計は、
もう1辺の距離より大きい
-

サンプル間の距離測定方法

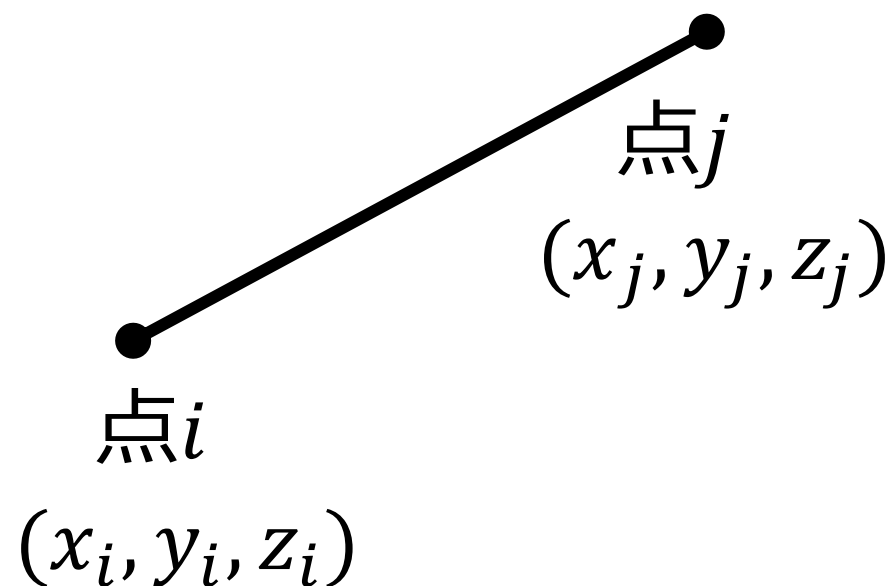
ユークリッド距離

$$\left[\sum_{k=1}^K (x_{ik} - x_{jk})^2 \right]^{1/2}$$

一番よく使われる L_2 距離とも言う
 K 次元空間の2点間の距離

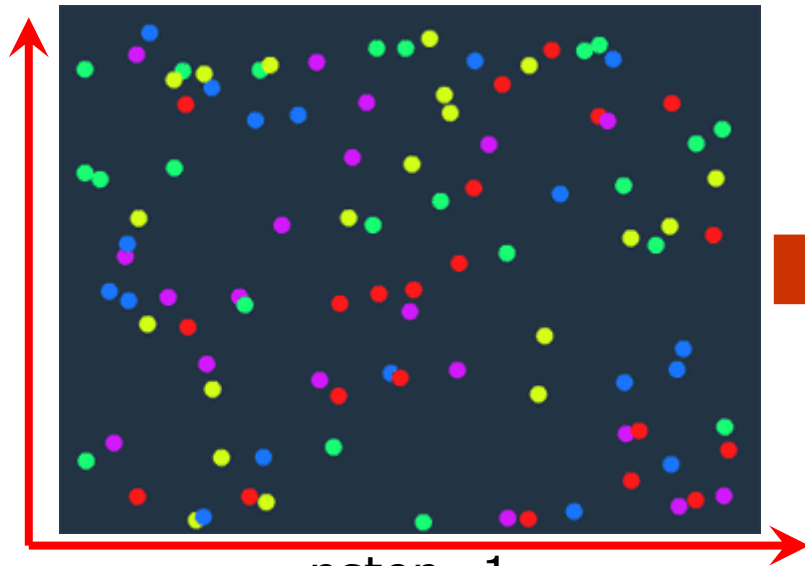
$$K = 3$$

例) 3次元

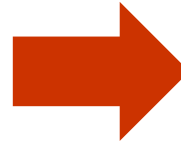


$$l = \left[(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \right]^{1/2}$$

k-means法



nstep=1



nstep=9



k-means法



メリット

ビッグデータの分析◎

デメリット

クラスタの数を
決めないといけない。
初期値に依存

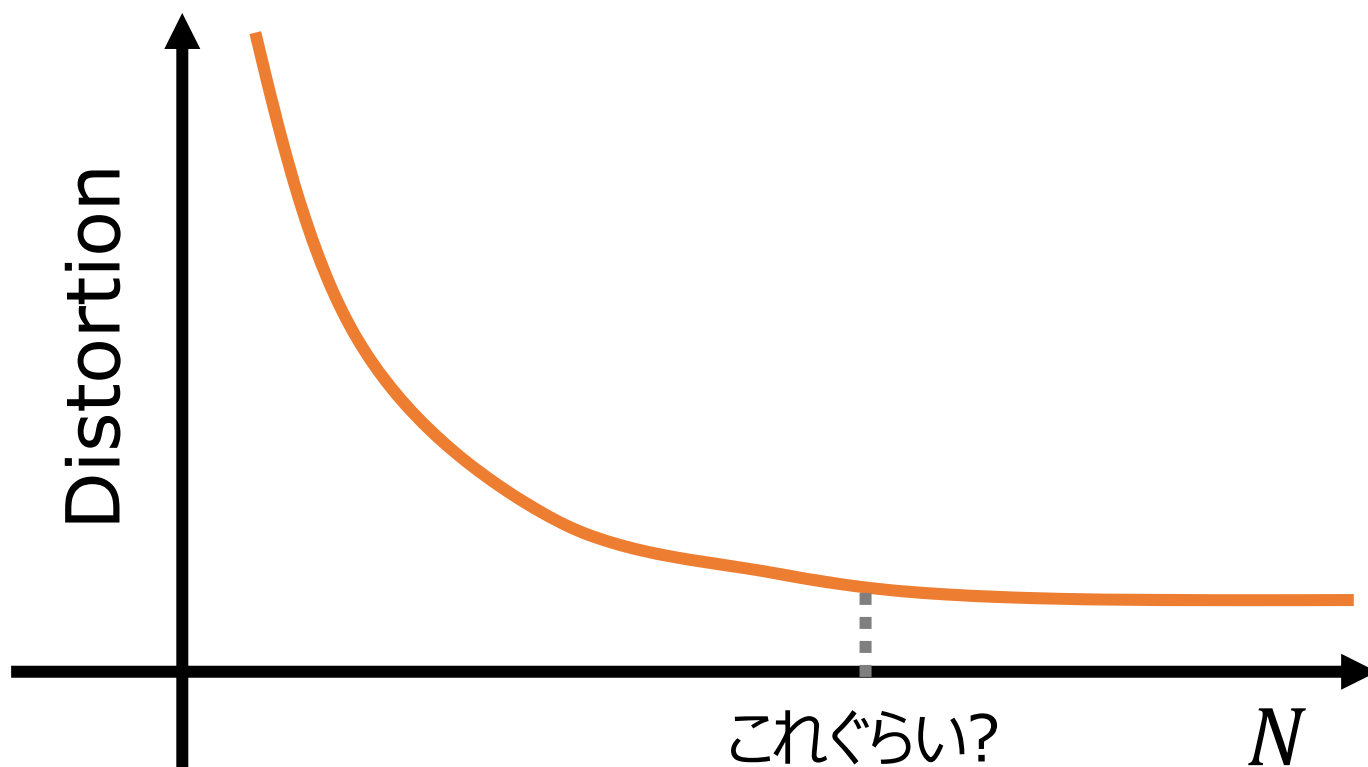


修正モデル

k-means++ 法
x-means法
が提案されている

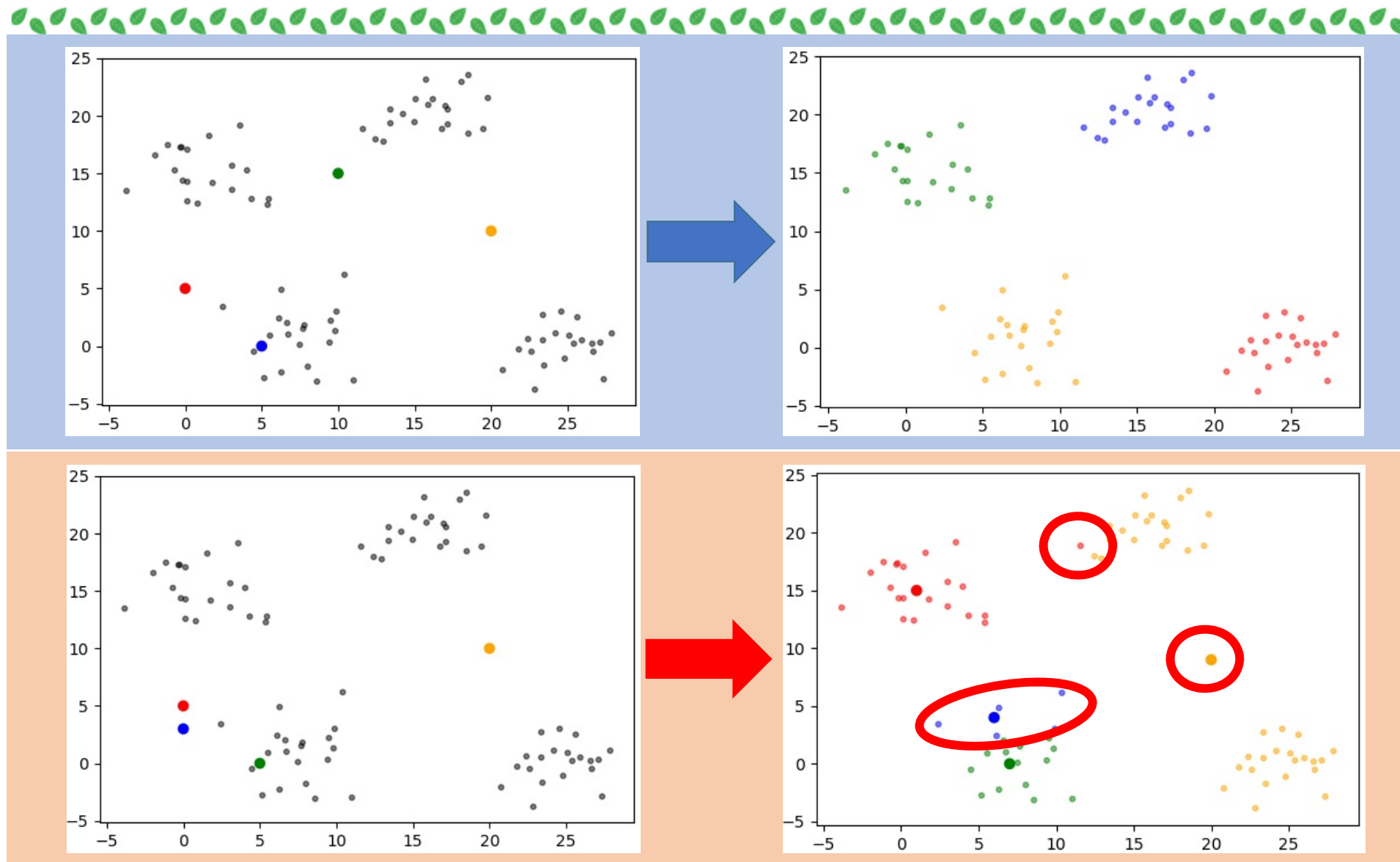
エルボー法

クラスタの重心点とクラスタ所属の各点の距離の総和



あまり減らなくなったところで決める

分析結果の初期値依存性



ヨビノリのk-means法の解説



■ヨビノリのk-means法の解説

<https://www.youtube.com/watch?v=8yptHd0JDlw>



■ k-means (k平均法)とは？【機械学習よくわかる用語集】

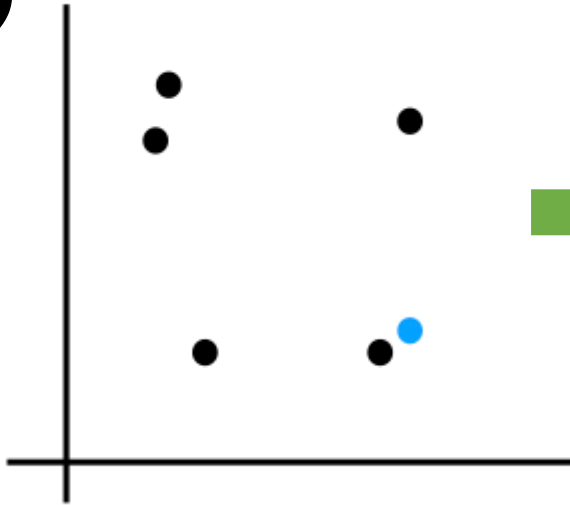
MATLAB Japan

<https://www.youtube.com/watch?v=Gg1xSvSY4YU>

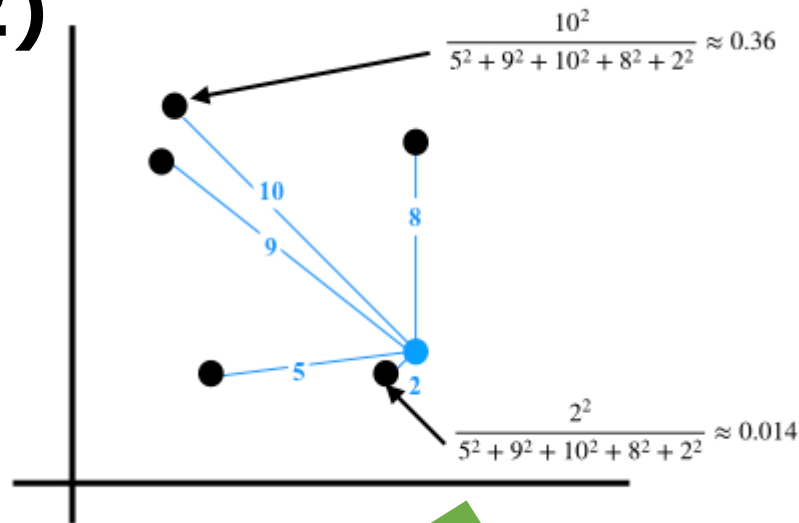


k-means++法

(1)



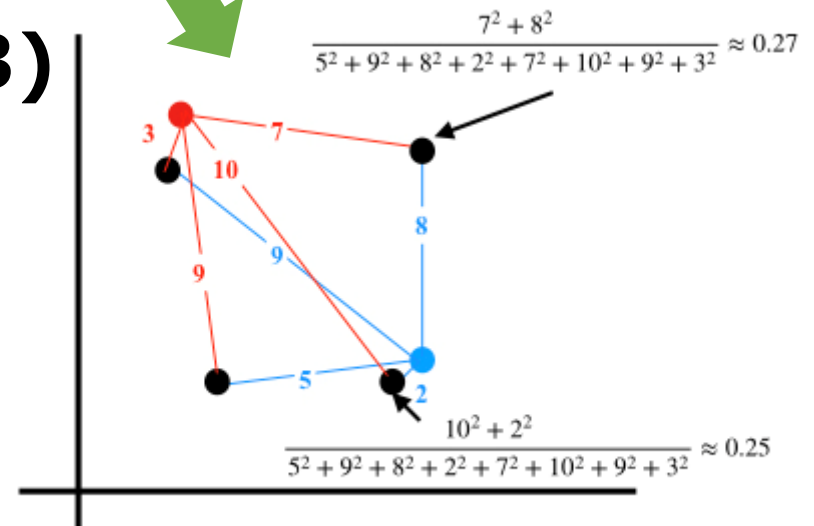
(2)



- (1) 1つ目の点はランダムに選ぶ。
- (2) 1つ目の点からできるだけ遠いところに2つ目の点を選ぶ。
- (3) 1つ目と2つ目からできるだけ遠いところに3つ目の点を選ぶ
- (4) k 個の点を選ぶまで続ける

その後の手法は全く同じ。

(3)



k-means++法

クラスタ中心からの距離

$$D(x)$$

次にクラスタ中心を選ぶ確率

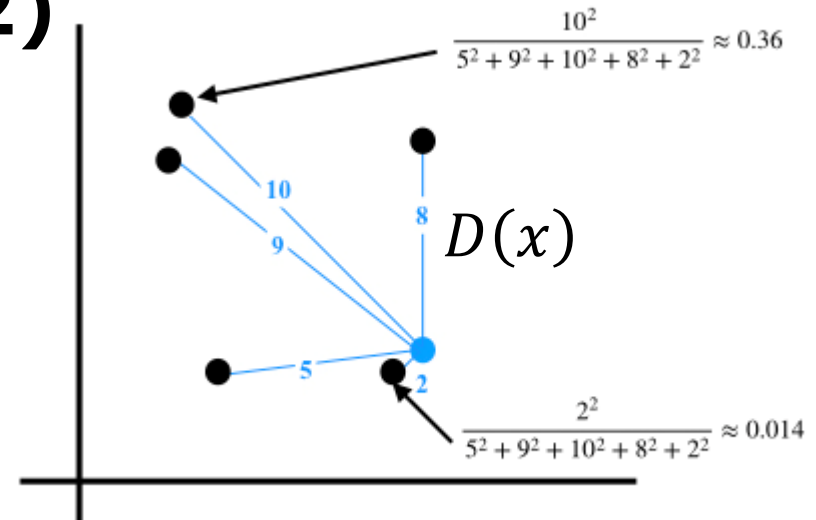
$$P = \frac{D(x)^2}{\sum D(x)^2}$$

2007年にDavid ArthurとSergei Vassilvitskiiによって提案

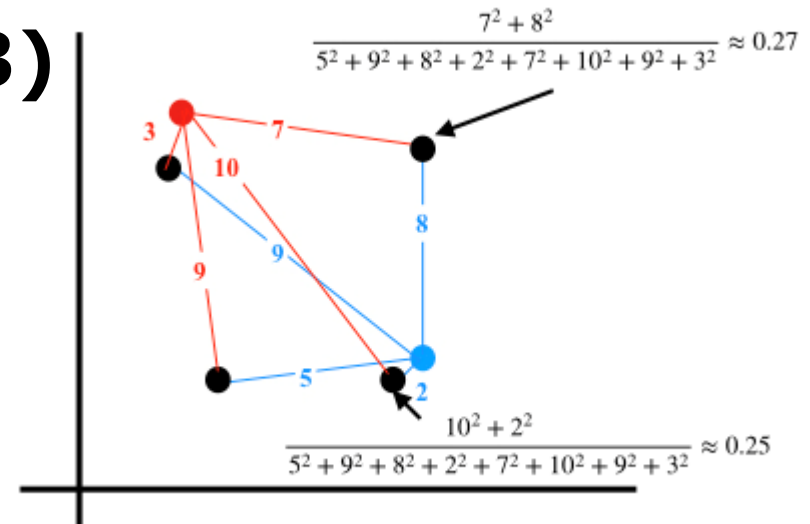
**k-means法に比べて、
収束スピードに関しては2倍
誤差が1000分の1**

from Wikipedia

(2)

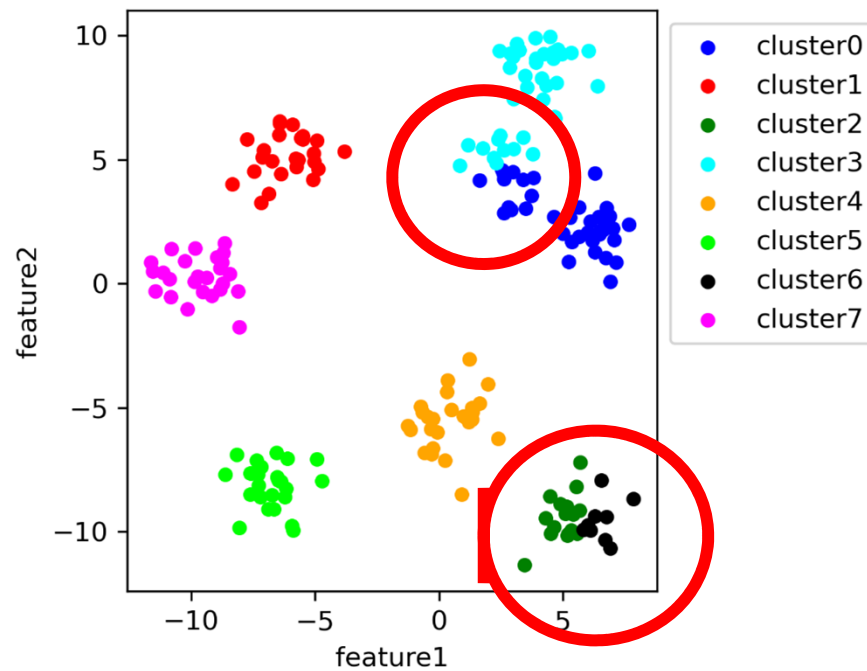


(3)

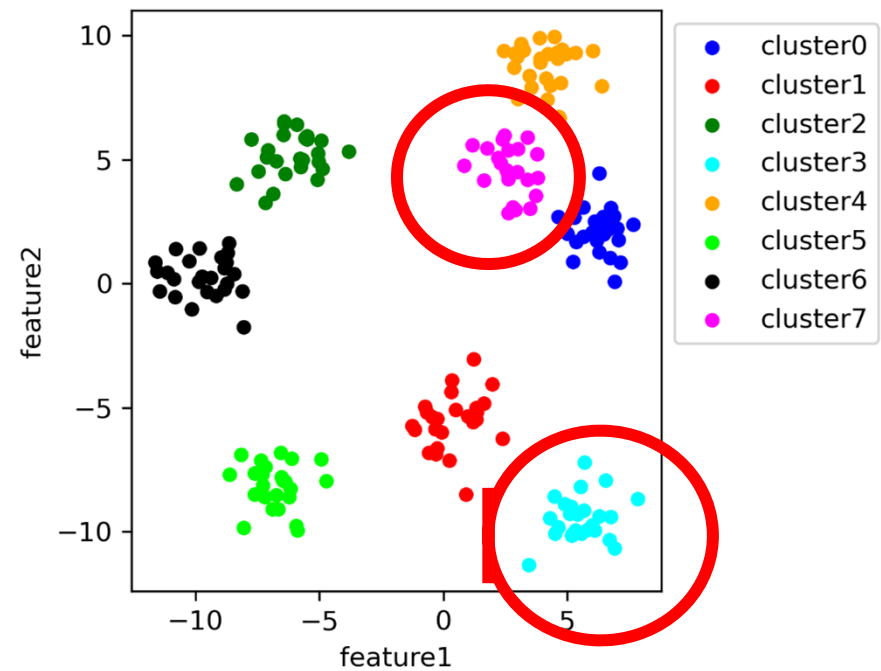


k-means vs. k-means++

k-means



k-means++



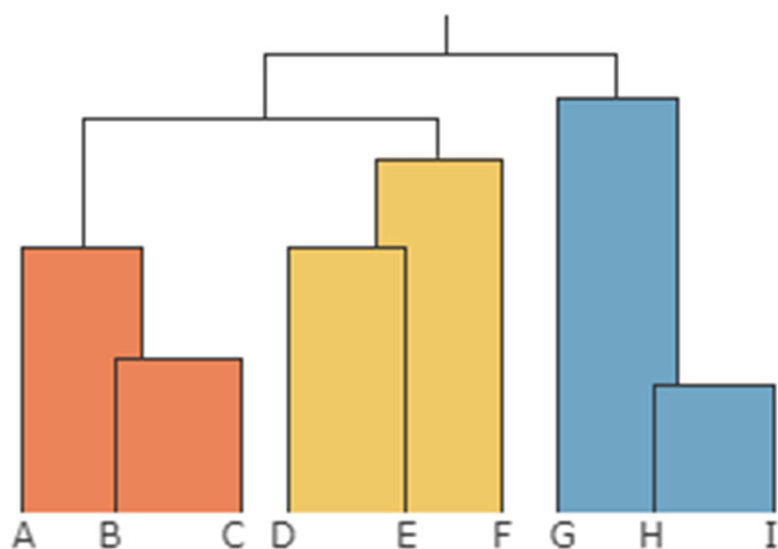
クラスター分析

東京大学のデータサイエンティスト育成講座 Ch.9-2

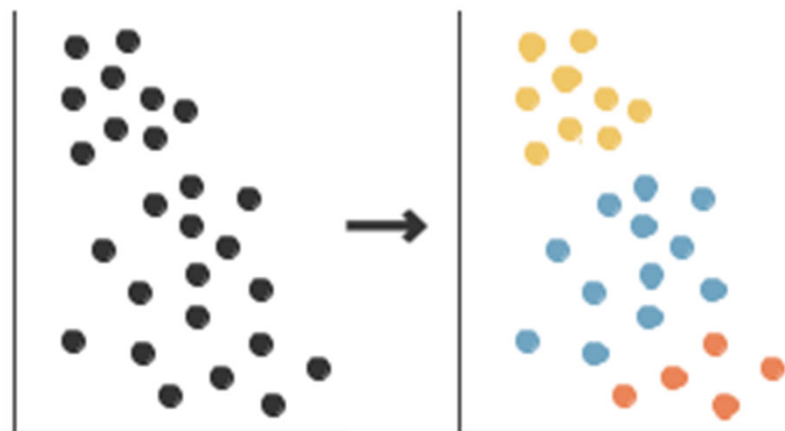
クラスター=Cluster

→房、集団、群れ

階層型クラスター分析



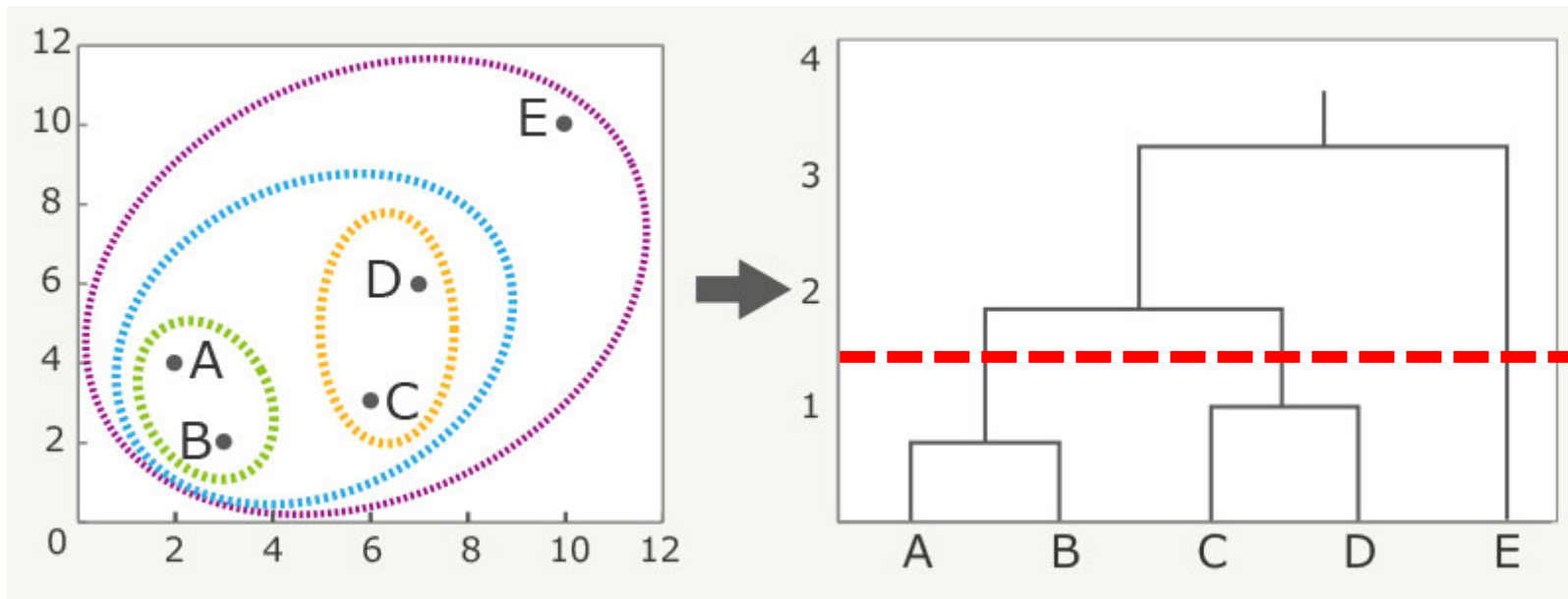
非階層型クラスター分析



https://promote.list-finder.jp/article/marke_all/cluster-analysis/

階層型クラスター解析

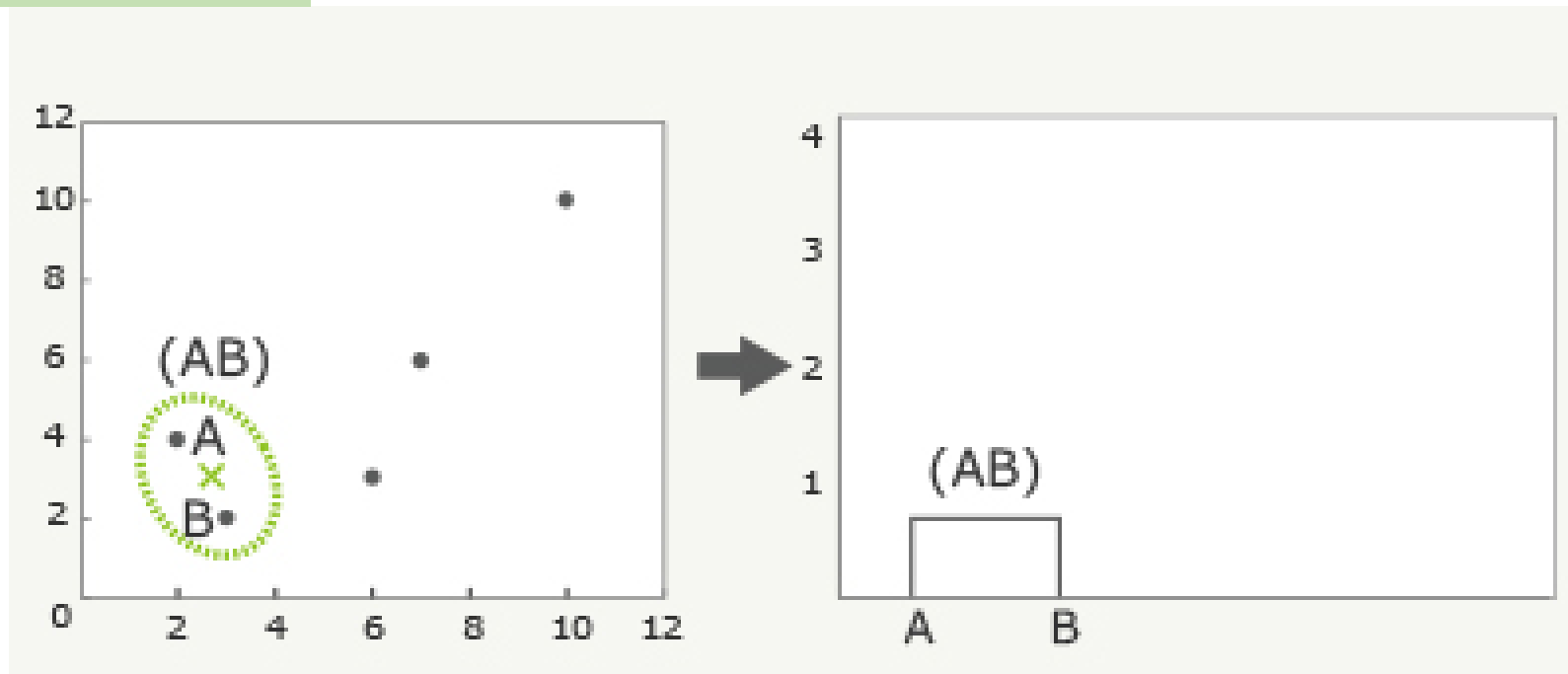
最も近いクラスターを1つずつ併合していき
クラスタリングを行う



- 階層型クラスター分析は結合されていく過程を一つひとつ確認できる
- **樹形図(dendrogram)**を任意の高さで切ることによって、欲しいクラスター数に分類できる。

階層型クラスター解析

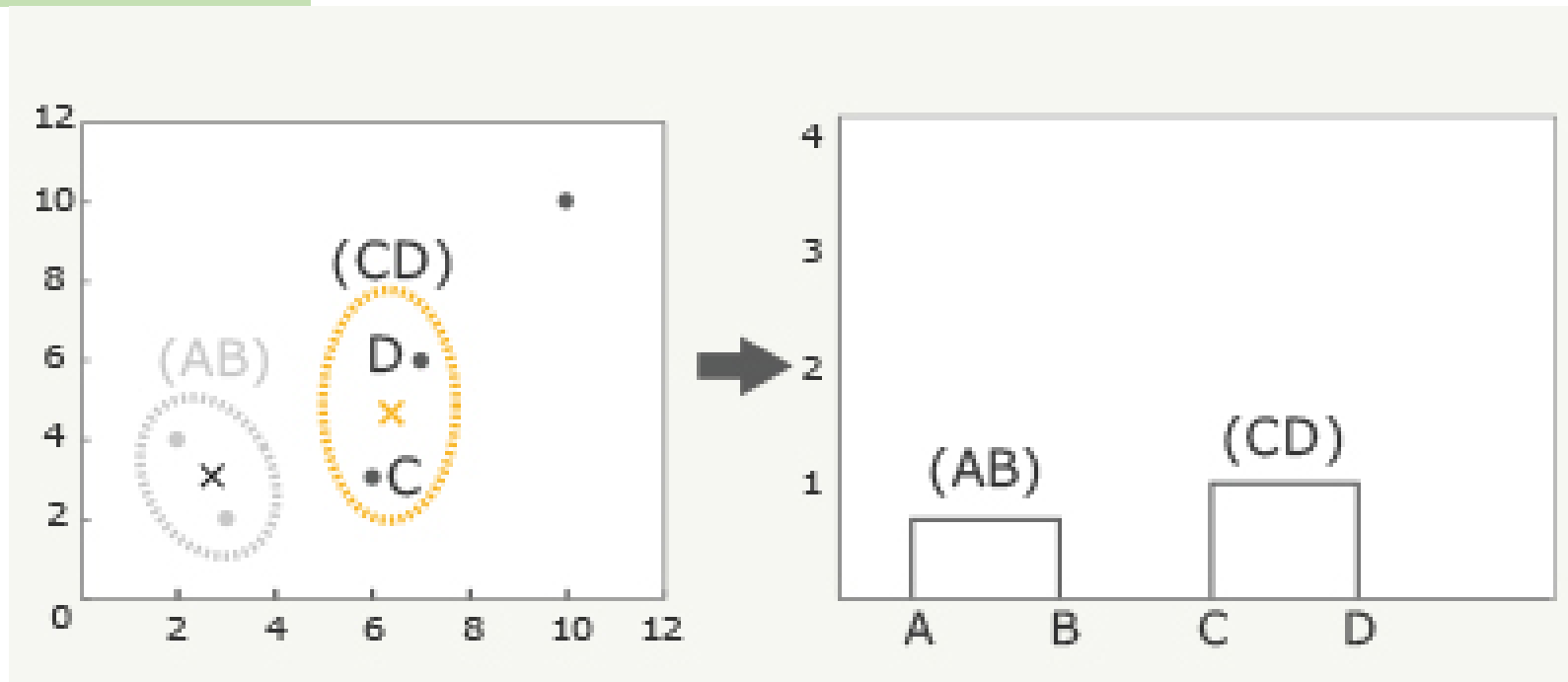
Step 1



A～Eの点で最も距離の近い組み合わせはAとBです。
そこで、まずはAとBをくくります。
次にこの2点の代表点（例えば重心）を求め、（AB）の×とします。

階層型クラスター解析

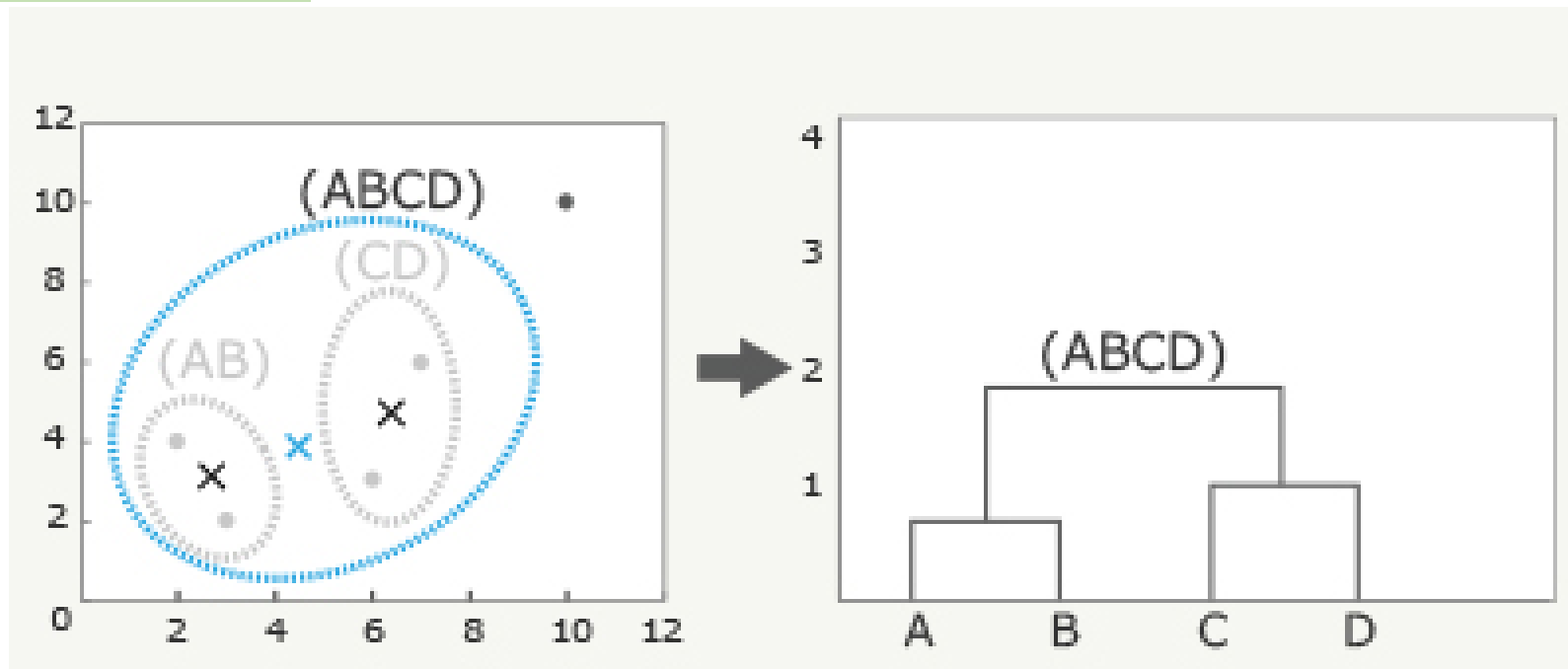
Step 2



(AB) の重心x、C、D、Eの4点で、最も距離の近い組み合わせを見つけます。ここではCとDが最も近いことが分かるので、CとDをくくります。この代表点を (CD) のxとします。

階層型クラスター解析

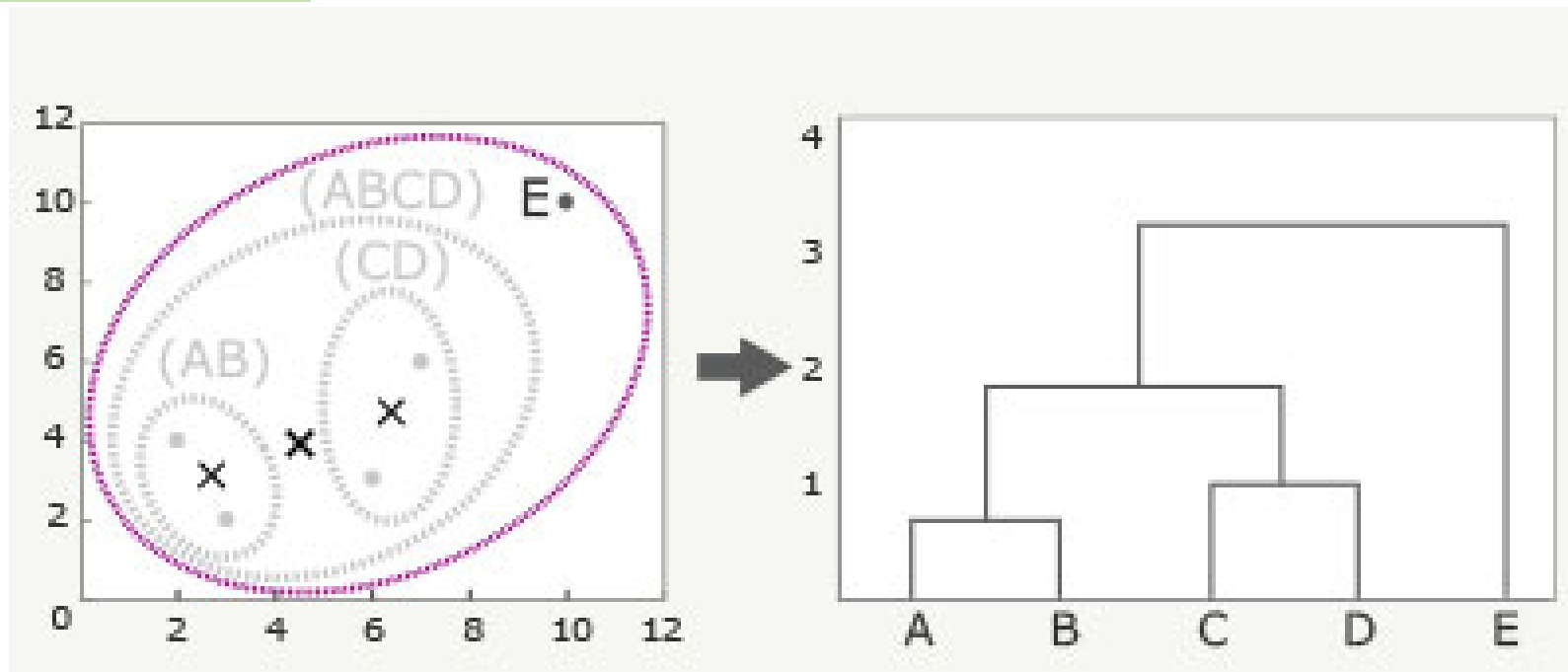
Step 3



(AB)、(CD)、Eの3点で最も距離の近い組み合わせを見つけます。
ここでは(AB)と(CD)が最も近いことが分かるので、(AB)と
(CD)をくくります。この代表値を(ABCD)の×とします。

階層型クラスター解析

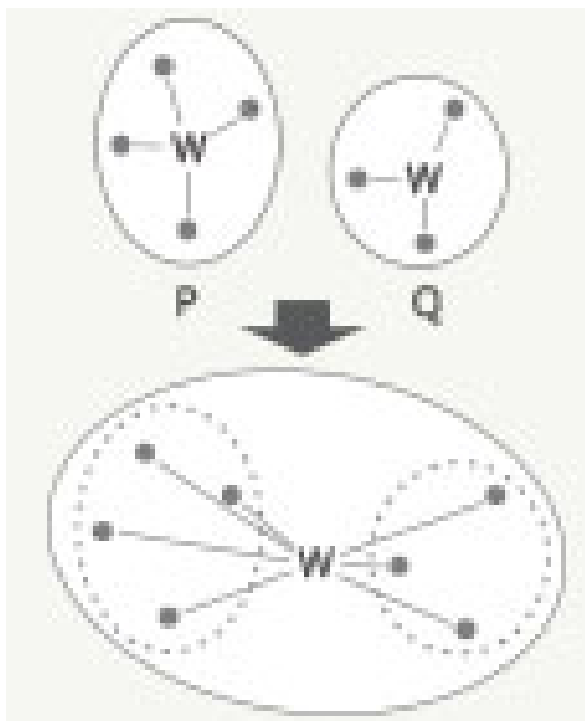
Step 4



最後にEをくくり樹形図にすると右図のようになります。
この時、AとBの上にある直線が、AとBの距離を表し、CとDの上にある直線がCとDの距離を表します。

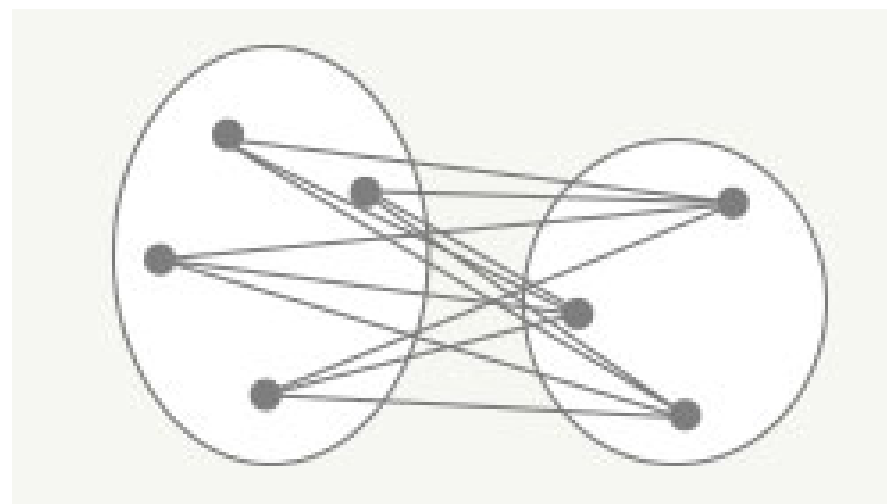
クラスタ間の距離測定方法

(1) ウォード法



→計算量は多いが分類感度がかなり良い。
そのため、よく用いられる。

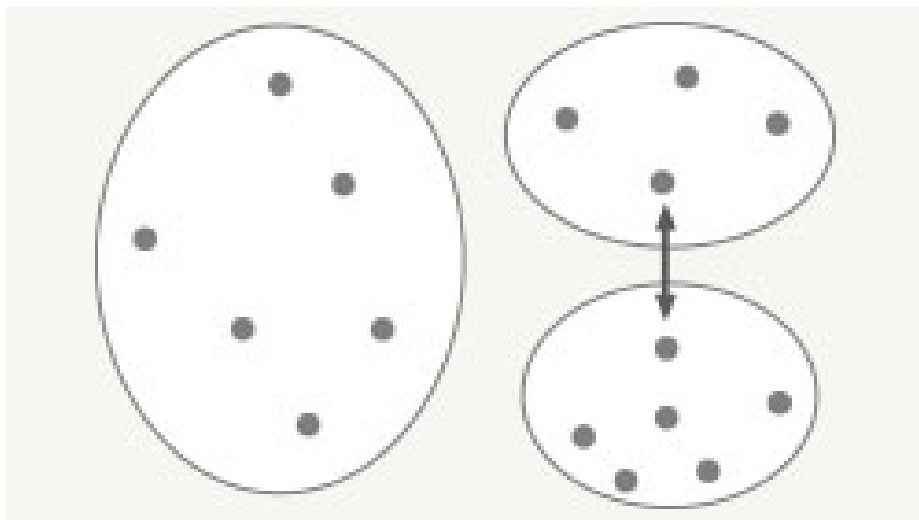
(2) 群平均法



各クラスター同士で、全ての組み合わせのサンプル間距離の平均をクラスター間距離とする手法。
→鎖効果や拡散現象を起こさないため、用いられることが多い。

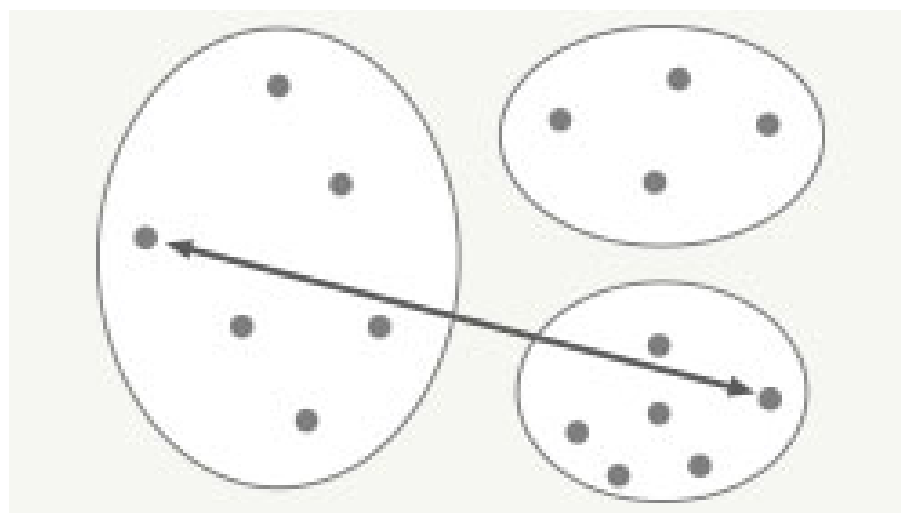
クラスタ間の距離測定方法

(3)最短距離法



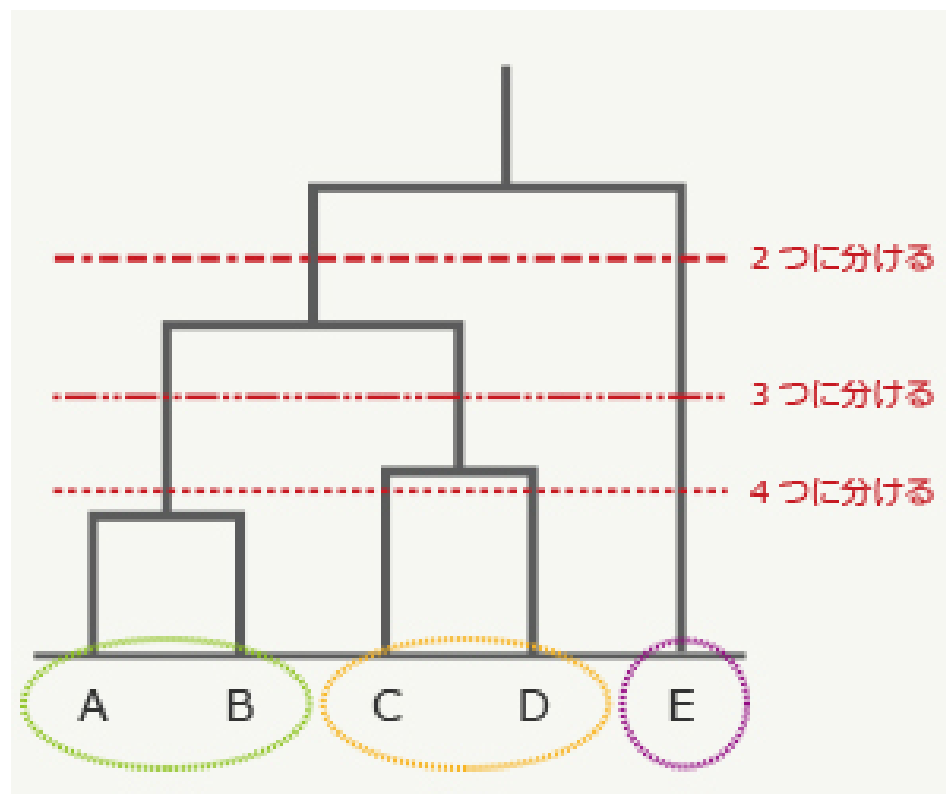
- ・ 2つのクラスターのサンプル同士で最も小さいサンプル間距離をクラスター間の距離とする手法。
- 鎖効果により、クラスターが帯状になってしまい、分類感度が低い。計算量が少ない。

(4)最長距離法



- ・ 最短距離法の逆で各クラスター中、最大のサンプル間距離をクラスター間距離とする。
- 分類感度は高いが、クラスター同士が離れてしまう拡散現象が生じる。計算量が少ない。

階層型クラスター解析



■ メリット

近いものから順番にくくるという方法をとるので、

あらかじめクラスター数を決める必要がないことが最大の長所。



■ デメリット

ビッグデータに不向き。

分析対象とするサンプルが膨大になると、分類が困難な場合も。


クラスタリング手法の比較



	階層型	非階層型
	樹形図を作る。 クラスタ間の距離の決め方 ウォード法, 群平均法, etc..	Hard:k-means法 Soft:混合ガウスモデル
ビッグデータへの 適応度		
クラスタ数	あとから自由に 変えられる (好きな大きさに分類できる)	基本的には 最初に決める (分類結果が初期に依存)

Scikit-learnで k-means法

k-means法解析の手順



1. scikit-learn のインポート
2. データの読み込み
3. headでデータの内容を確認
4. shapeでデータのサイズを確認
5. isnulで欠損値の確認→あれば削除(補完など。。)
6. インスタンスを作成
7. .fitを実行
8. .predictを実行
9. 可視化

これはいつもの手順

大事なのはここ!!

(1) scikit-learn のインポート



```
from sklearn.cluster import KMeans
```



Numpy, Pandasの時と一緒に

(2) データの読み込み

乱数を使って
サンプルデータを
生成することもある。

■ CSV ファイルの読み込み

```
df = pd.read_csv('data/w5_rep_lattice.csv')
```

1 列目をインデックスに入れるかどうか、オプションで指定できます。

<https://note.nkmk.me/python-pandas-read-csv-tsv/>

(2) データの読み込み

■ Scikit-learnのサンプルデータの読み込み

アヤメのデータや、乳がんのデータなど、いろいろあります。

```
from sklearn import datasets  
iris = datasets.load_iris()
```

このままだとirisはBunch型という
変数になります。

```
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
```

試しにtype(iris) と実行してみましょう。
Bunch型のままでも解析できるのですが、
DataFrameで統一しました。

そのため、この行を使って、
DataFrameに変換しています。

データの構成



■ **iris.data**: 説明変数（アヤメのデータ）

- sepal length: 花のがくの長さ
- sepal width: 花のがくの幅
- petal length: 花弁の長さ
- petal width: 花弁の幅

■ **iris.feature_names**: 説明変数のindex

■ **iris.target** : 目的変数(アヤメの種類) → 答え合わせに使う

(3) headでデータの内容を確認


df.head()

で、データの中身を確認。

```
1 iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
2 iris_df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

(4) shapeでデータのサイズを確認




df.shape

でDataFrameのサイズを確認。

df.isnull()

でそれぞれの列にいくつNaNがあるか確認。



(5) 欠損値の確認→あれば削除



それぞれの列にいくつNaNがあるか

df.isnull().sum()

欠損値が一つでもあれば、その行を削除

df = df.dropna(how='any')

本当は、この後、

df.isnull().sum()

をもう一回やって、欠損値が本当になくなったことを確認したほうがいい。

df.shape

をもう一回やって、欠損値削除後のデータサイズも確認したほうがいい。



欠損値の取り扱い@Pandas

~~~~~

データに欠けてる値(空白や異常値)があった場合  
ファイルをまとめて分析する前に、  
データ処理を行います。

**\* 欠損値があるか確認 / Check if there are NaN**



**\* 補完 / Complement**

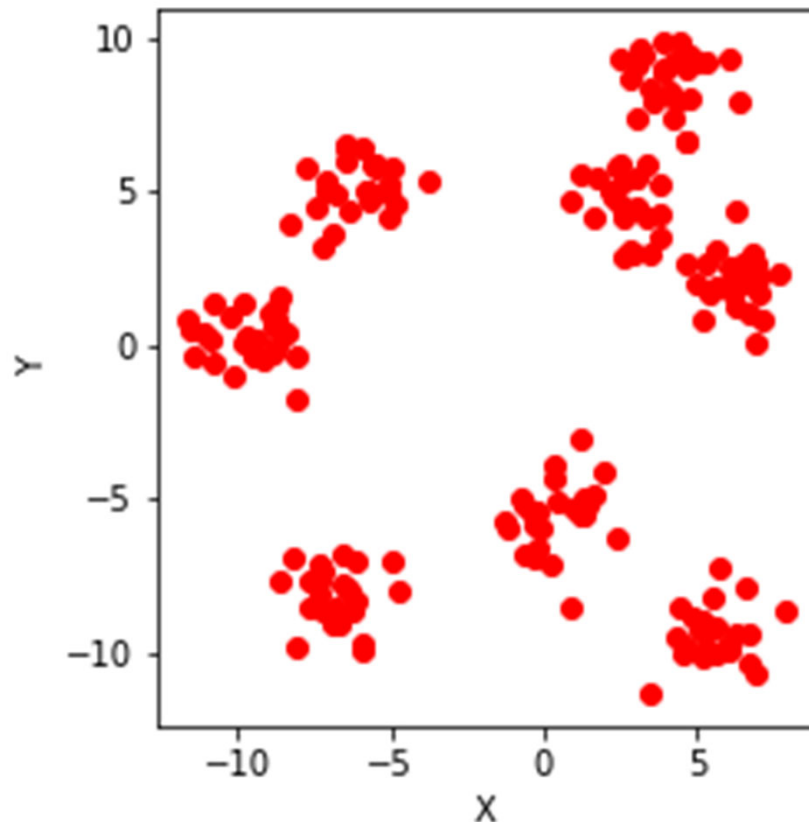
**\* 置換 / Replace : 0や平均値で置換**

**\* 抽出 / Extract**

---

## (6) データを可視化してチェック

いくつかのクラスタに分けたらいいか、  
プロットして確認する

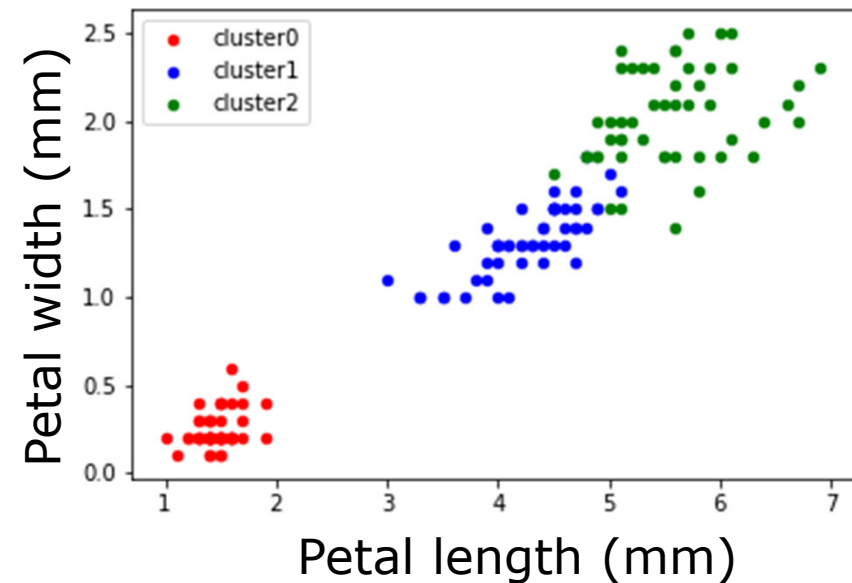
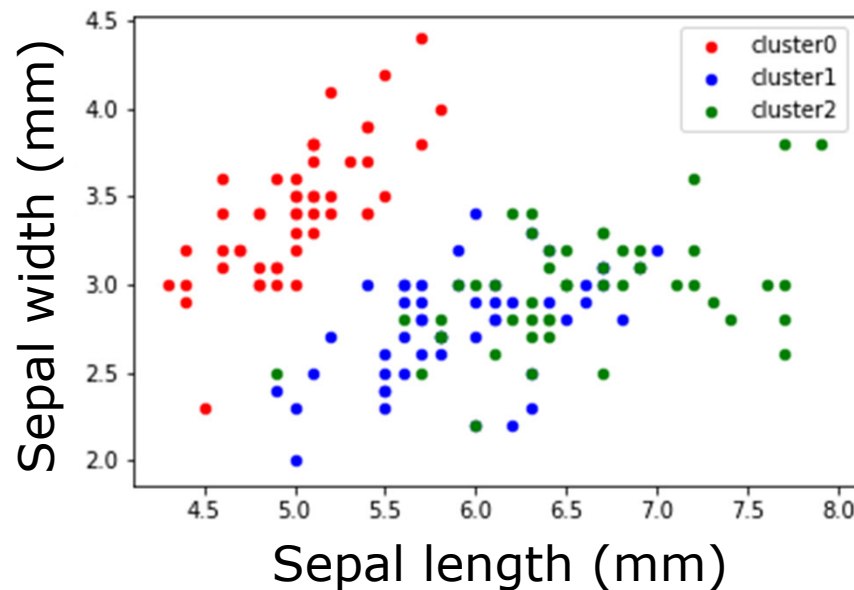


k-means法では、  
クラスタの数を最初に  
自分で決めないといけない。  
だいたいのあたりを付けて  
おこう。

## (6)データを可視化してチェック



- 0: Setosa
- 1: Versicolor
- 2: Versinica



4次元のデータを2次元面内に射影して試しているので、  
2変数で見たときに混ざっているからと言って、使わないほうがいいとも限らない。  
いろいろな組み合わせで、うまく分類できるかどうか試してみるとよい。  
答えがない場合も、ぼんやりクラスターが見えるかどうか、可視化して確認する。

## (7)インスタンスを作成



```
kmeans = KMeans(n_clusters=3, init="random")
```


オプション

- n\_clusters: クラスタの数
- max\_iter: 学習のループ回数
- init: 平均の初期値の決め方
- n\_jobs: k-meansを何並列にするか(-1ならばpcのコア数分だけ並列してくれます)

**クラスタの数を最初に  
自分で決めないといけない**

オプションの詳細はこちら（演習問題からもリンク張ってあります）

<https://pythondatascience.plavox.info/scikit-learn/%E3%82%AF%E3%83%A9%E3%82%B9%E3%82%BF%E5%88%86%E6%9E%90-k-means>



## (8) .fitを実行



#k-means法を実行

```
kmeans.fit(X)
```

Xにデータを入れます。

### 大事なポイント!!!

DataFrameのままだと.fitに入れられないので、.valuesを使って、Numpy arrayに変換しておきます。

クラスタリングは、使う変数を自分で選んでもよい。

```
kmeans.fit(iris_df.values[:,0:4])
```






## (9) .predictを実行



# クラスタ番号を予測 / Predict the cluster number.

```
y_pred = kmeans.predict(X)
```

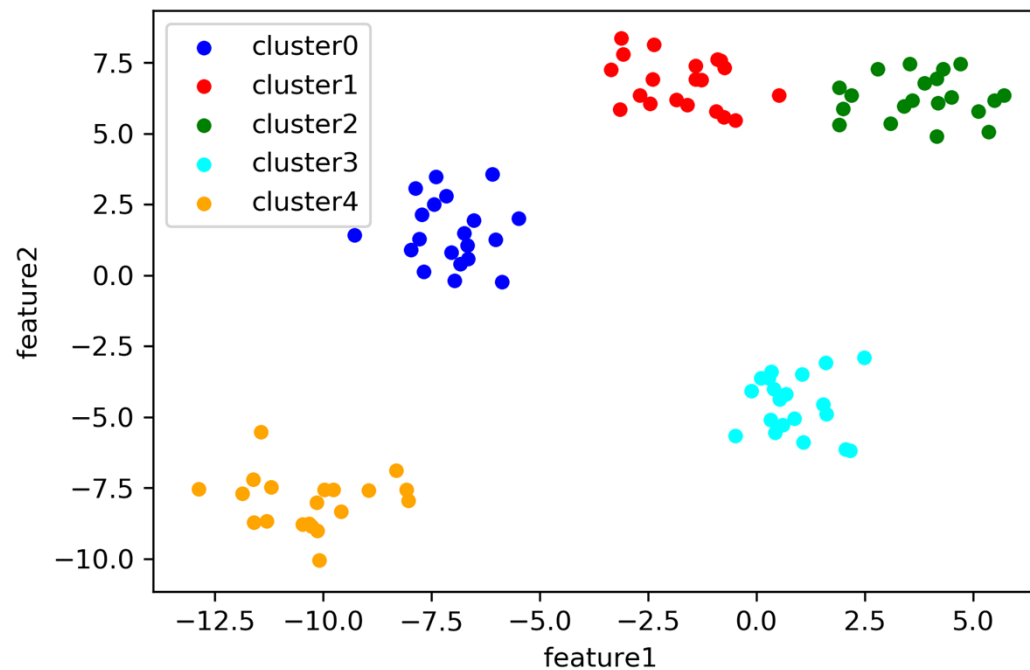


Xにデータを入れます。

---

# (10) 解析後：可視化

分類したデータを可視化して確認  
→ クラスタごとに色分けしてプロット



# k-means法解析の手順

1. scikit-learn のインポート
2. データの読み込み
3. headでデータの内容を確認
4. shapeでデータのサイズを確認
5. isnullで欠損値の確認→あれば削除(補完など。)
6. 可視化
7. インスタンスを作成
8. .fitを実行
9. .predictを実行
10. 可視化

これはいつもの手順

大事なのはここ!!